AD-A269 627

MENTATION PAGE

Form Approved
OBM No 0704-0188

ed to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and
ction of information. Send comments regarding this burden or any other aspect of this collection of information, including suggestions
irectorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to
ect (0704-0188), Washington, DC 20503.

...port Date. 1992	3. Report Type and Dates Covered. Final - Journal Article
4. Title and Subtitle. Implementation of Custom Colors in the DECwindows Environment	5. Funding Numbers. *Program Element No* 0604262; *Project No* 64214; *Task No*; *Accession No* DN257017; *Work Unit No* 93512F
6. Author(s). Stephanie A. Myrick, Maura C. Lohrenz and Perry B. Wischow	
7. Performing Organization Name(s) and Address(es). Naval Research Laboratory, Mapping, Charting and Geodesy Branch, Stennis Space Center, MS 39529-5004	8. Performing Or...ization Report Number. NRL/JA 7441--92 0007
9. Sponsoring/Monitoring Agency Name(s) and Address(es). Naval Air Systems Command, Department of the Navy, Washington, DC 20361-8030	10. Sponsoring/Monitoring Agency Report Number. NRL/JA/7441--92-0007
11. Supplementary Notes. Published in Transactions of DECUS.	
12a. Distribution/Availability Statement. Approved for public release; distribution is unlimited.	12b. Distribution Code.

13. Abstract *(Maximum 200 words)*.

This paper describes the implementation of user-defined, or custom, colors in the DECwindows environment. Custom colors can be used to augment the standard color set that is associated with the hardware colormap. The custom color set that is included in this paper consists of 240 distinct colors, each of which is comprised of red, green, and blue intensities. Intensity levels range from zero (no intensity) to 255 (maximum intensity). The DECwindow and Xwindow statements, which are required to implement the custom color set, are identified and described. Some of the tasks performed by these statements include creating and installing a colormap and accessing individual colors. Using a VAX C programming language platform, examples of proper invocation and syntax of these statements are provided. Some familiarization with C, DECwindows and Xwindows is assumed.

9

93-22000

14. Subject Terms. Digital maps, optical storate, databases, data compression			15. Number of Pages. 7
			16. Price Code.
17. Security Classification of Report. Unclassified	18. Security Classification of This Page. Unclassified	19. Security Classification of Abstract. Unclassified	20. Limitation of Abstract. SAR

NSN 7540 01 280 5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Implementation of Custom Colors in the DECwindows Environment

Stephanie A. Myrick, Maura C. Lohrenz And Perry B. Wischow
Mapping Sciences Branch, Naval Research Laboratory
Stennis Space Center, MS

Abstract

This paper describes the implementation of user-defined, or custom, colors in the DECwindows environment. Custom colors can be used to augment the standard color set that is associated with the hardware colormap. The custom color set that is included in this paper consists of 240 distinct colors, each of which is comprised of red, green, and blue intensities. Intensity levels range from zero (no intensity) to 255 (maximum intensity). The DECwindow and Xwindow statements, which are required to implement the custom color set, are identified and described. Some of the tasks performed by these statements include creating and installing a colormap and accessing individual colors. Using a VAX C programming language platform, examples of proper invocation and syntax of these statements are provided. Some familiarization with C, DECwindows and Xwindows is assumed.

INTRODUCTION

The Naval Research Laboratory Detachment at the Stennis Space Center (NRL-SSC) is developing a database of scanned aeronautical chart images, the Compressed Aeronautical Chart (CAC), for use in aircraft digital moving-map systems and for mission planning (Lohrenz and Ryan, 1990). The CAC uses a set of 30 custom color palettes. Each palette consists of 240 distinct colors, and each color is comprised of RGB intensities. Intensity levels range from 0 (no intensity) to 255 (maximum intensity). Although CAC is primarily destined for video display within the aircraft, the need to display CAC within the laboratory (primarily for quality control) also exists.

The CAC database is comprised of scanned and compressed aeronautical chart images at six different scales, where each scale represents a different chart series (Figure 1). The original paper charts and their scanned, digital equivalents are distributed by the Defense Mapping Agency (DMA). Each series of DMA paper charts is depicted by a different set of ink colors. Sometimes, different charts within a single series may utilize different inks since, for example, not all DMA charts in a given series are produced by the United States. Different countries have different chart production standards, including ink colors. When DMA scans the paper charts into digital form, the original chart colors are retained as red, green and blue (RGB) intensities. More than 16 million possible colors exist in this scanned data set; each color is represented by 24 bits (DMA, 1989)

The CAC database uses only 8 bits to represent each color (Lohrenz, 1991); 24-to-8 bit color palettes are used to map each 8-bit value to its full, 24-bit, RGB intensities. Therefore, the number of possible chart colors are reduced from over 16 million (in the original scanned chart database) to 256 (in a given CAC color palette). Sixteen palette colors are reserved for vector overlay data, so each CAC color palette contains only 240 colors with which to display CAC data. During the compression of scanned chart data into CAC, all of the chart data at a given scale (chart series) are partitioned into 5 latitude-based geographic zones (Figure 2). Each zone uses one standard CAC color palette. Six chart scales and five zones per scale result in 30 unique CAC color palettes. Since each palette contains 240 distinct colors, there are up to 7200 possible colors in the CAC database, which represents a significant reduction from the 16 million colors in the original database.

One method of presenting color data on a limited-color display device involves matching each color in the data set to the most representative color in the default color set for that display device. This is not practical in the case of the CAC database: there are simply too many possible colors (7200) to be matched. A more reasonable approach is to

Chart Scale	DMA Source Chart
1:2,000,000	Jet Navigation Chart
1:1,000,000	Operational Navigation Chart
1:500,000	Tactical Pilotage Chart
1:250,000	Joint Operational Graphic
1:100,000	Topographic Line Map-100
1:50,000	Topographic Line Map-50

Figure 1. Chart scales and source charts

create a display colormap that can be used with the appropriate CAC color palette.

NRL-SSC has developed a software suite to display

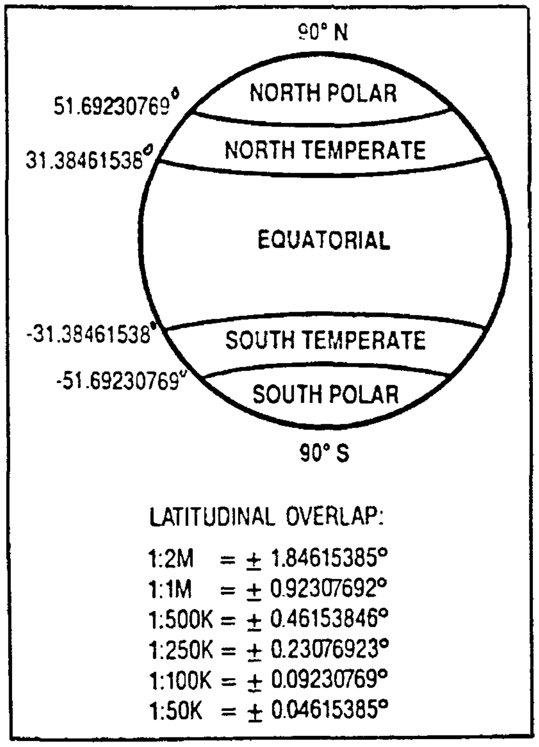

Figure 2. Geographic zones with latitudinal boundaries

compressed chart data from the CAC database. This software is written in the C programming language and utilizes DECwindows and X11 window application software. Each program within the suite utilizes a common subprogram, named LOADCOLORMAP_X.C, to load the representative colormaps. This paper presents the programming statements that are executed within LOADCOLORMAP_X.C.

APPLICATION SOFTWARE

Separate application software packages exist for DECwindows and Xwindows. If the program is to be executed on a system running under VMS, then it is appropriate to apply the DECwindows toolkit. non-VMS systems will use the X11 applications toolkit. The statements in Table 1 are used to determine the appropriate application toolkit.

```
# define VMS  /* This definition must be
             /* declared elsewhere,*/
            /* when appropriate. If */
           /*defined, compilation  */
          /* will build for a VMS
         /*DECwindows system.    */
#ifdef VMS
#include <decw$include/DwtAppl.h>
         /* DECwindows Toolkit      */
#else
#include <X11/DwtAppl.h>
       /* X11 DECwindows Toolkit */
#endif
```

Table 1. Determining Appropriate Application Toolkit

VARIABLE DECLARATIONS

Variables are kept localized whenever possible. Hence, LOADCOLORMAP_X.C declares the following variables, as listed in Table 2.

COLORMAPS

A colormap is used to translate each pixel's bit value into a color. Figure 3 (from Nye, 1988a) illustrates how a colormap is used to map an 8-bit pixel value to a colorcell (i.e., an RGB color). Color displays use multiple bits per pixel (bit-planes) to specify colors. The number of different colors that can be displayed on a monitor is dependent on the number of bit-planes that are supported by the monitor. For example, a four-plane system could index 2^4 colorcells, or 16 distinct colors. Similarly, an eight-plane system could index 2^8 colorcells, or 256 distinct colors. Since CAC uses up to 240 colors, only monitors with at least 8 bit-planes can

```
#define  MAX_COLORS 256  /* Maximum number of CAC colors.*/

/********************************************/
/* Define colormaps, visuals, Graphics Contexts, etc. */
/********************************************/
static Visual       *defVisual;      ! Points to the default
                                      ! visual

static XVisualInfo  *visualList;     ! Points to a visual
                                     ! structure

static XVisualInfo  visualTemplate;  ! Specifies attributes
                                     ! to be matched

static int          num_visuals;     ! Number of visual
                                     ! structures
static Colormap     colormap;        ! The hardware color
                                     ! map

static int         colorcount;       ! The number of CAC colors

static short       colormap_size;    ! Size of the hardware
                                     ! colormap

static XColor      *colorcell_defs;  ! XColor definition
                                     ! structure

static unsigned long  *colors;       ! Points to XColor
                                     ! definition structure

static Display   *colormap_display;  ! Display using CAC
                                     ! colors
static Window     colormap_window;   ! Window using CAC
                                     ! color
```

Table 2. Variable Declarations For Subprogram Loadcolormap_X.C

up to 240 colors, only monitors with at least 8 bit-planes can display CAC data in its intended form. CAC can be displayed on monitors with fewer than 8 bit- planes, but the quality of the colors will be degraded. The degree of color degradation depends on how well the 240 CAC colors can be remapped to the monitor's smaller color set (Trenchard, in preparation).

The most common type of display contains between 4 and 8 bit-planes and uses the above-mentioned colormap indexing method. These are known as mid-range displays. The colorcells may be either static (i.e., cannot be modified) or read/write (can be filled with arbitrary RGB values).

High-performance displays use up to 24 bit-planes and can display 16 million distinct colors. With so many colors, the above-mentioned indexing technique would not work efficiently. Therefore, high-performance displays use three separate colormaps, one colormap for each primary, as shown in Figure 4 (from Nye, 1988a).

VISUAL CLASS

A visual structure, which contains information about the display, must be specified when the colormap is created. Information such as the number of colorcells and the type of visual class is available through the visual structure. Because the hardware colormap will be modified to accommodate CAC colors, only read/write visual classes can be used. There are several different visual classes that apply to read-/write display types, including GrayScale, PseudoColor and DirectColor (Nye, 1988a). Because CAC requires the use of color, the GrayScale visual class is inappropriate. However, the PseudoColor and DirectColor visuals will accommodate

```
/*Initialize a count for the maximum number of custom colors*/
  colorcount = MAX_COLORS;

 /* Identify the colormap display and window areas */
 colormap_display = display;
 colormap_window  = window;

/* Get the default visual */
 defVisual = DefaultVisual(colormap_display,0);

 /* Use the default visualid for our template */
 visualTemplate.visualid = defVisual->visualid;

/* Get the supported visuals (Nye, 1988b) */
 visualList = XGetVisualInfo
            (colormap_display, ! Pointer to Display structure
             VisualIDMask,     ! Visual mask, elements to match
             &visualTemplate,  ! Attributes to use in matching
             &num_visuals)     ! Returns number of matching
                               ! visual structures

 /* This application must use either PseudoColor */
 /* or DirectColor. */
 if (visualList -> class != PseudoColor &&
     visualList -> class != DirectColor)
   {
     printf("Unable to use this Visual Class...\n");
     exit (0);
   }

/* Determine that an adequate number of planes */
/* is available to allow for 256 colors.      */
 if (visualList->colormap_size < colorcount)
   {
    printf("This visual must support at least %d colors...
            \n",colorcount);
   }
```

Table 3. Visual Class Determination

CAC data and can be used. Table 3 presents the statements that are used to determine a visual class.

LOADING THE COLORMAP WITH CAC COLORS

Table 4 presents the statements that modify the display hardware colormap to use CAC colors. The statements accomplish the loading of CAC colors in three basic steps. First, memory is allocated for use with **colors**, which is a pointer to the Xcolor definition structure, **colorcell_defs**. **Colors** serves an intermediate purpose, in that it is used to map the Xcolor definition structure to actual hardware colormap colorcell addresses. In the second step, the Xcolor definition structure, **colorcell_defs**, is loaded with individual CAC colors. This structure is used to define the RGB components of each CAC color. The third step modifies the hardware colormap by storing (i.e. overwriting) the colormap's individual colorcells with CAC RGB colors, as defined by **colorcell_defs**. The relationship between **colors**, **colorcell_defs** and the hardware colormap is shown in Figure 5.

Figure 5 also illustrates that the display hardware colormap is modified, beginning at some memory location (here, at location 8), to contain CAC colors. In order to access these colors, this hardware colormap offset (which accommodates the custom colors) must be used. The hardware

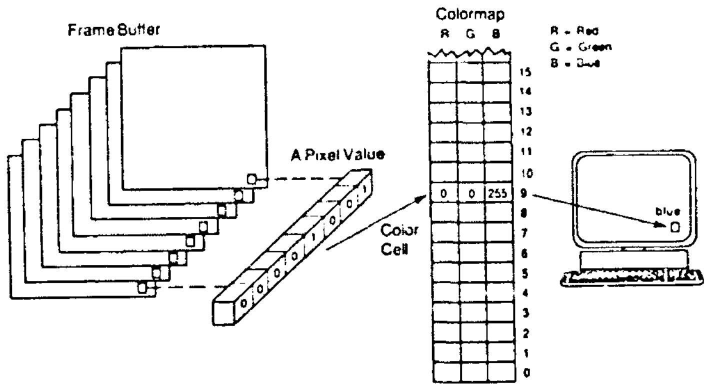

Figure 3. Pixel value to RGB mapping (Nye, 1988a)

colormap offset is found within **colors[0]**. Therefore, if the following statement is used to obtain the offset:

colormap_offset = colors[0];

then the following statements can be used to adjust CAC data values to the appropriate hardware colormap address:

for (i=0; i<=numpoints; i++)

CAC_data_buffer[i] += colormap_offset;

SUMMARY

Due to the large number of individual colors, custom CAC colors are used to augment the standard color set that is associated with the hardware colormap. Augmenting the hardware colormap is a more reasonable approach than trying to match all of the individual colors to the default color set.

The DECwindow and Xwindow statements that are required to implement custom colors have been provided and described. These statements incorporate CAC palette colors into the default hardware colormap for displaying CAC

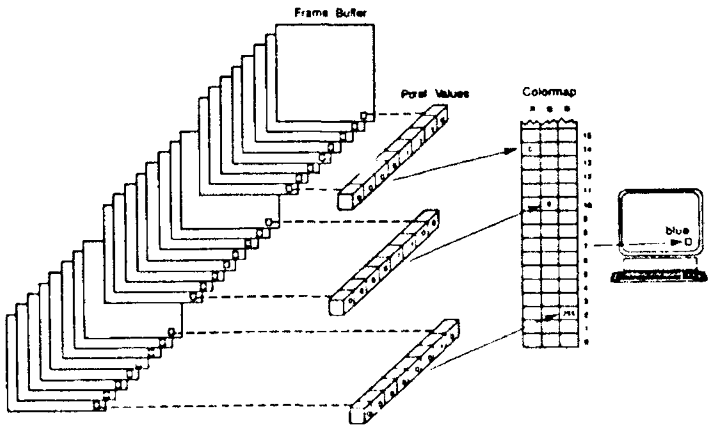

Figure 4. Pixel value to RGB mapping: high performance color displays (Nye, 1988a)

```
/* Associate the default colormap with the colormap display. */
 colormap = XDefaultColormap
            (colormap_display,
             DefaultScreen(colormap_display));

 /* Associate the default colormap with the specified window  */
 /* (Nye, 1988b).                                             */
 XSetWindowColormap
    (colormap_display, ! Pointer to Display structure
     colormap_window,  ! Window id, for which to set the colormap
     colormap);        ! Specifies the colormap

 /* Allocate enough memory to accommodate CAC custom colors. */
 colorcell_defs = (XColor *)
                  XtMalloc(sizeof(*colorcell_defs)*colorcount);

 /* Allocate space for the XAllocateColorCells call */
 colors = (unsigned long *)XtMalloc(sizeof(*colors)*colorcount);

 /* Allocate colors for use with the hardware colormap */
 /* (Nye 1988b).  See figure 5.                         */
  if (!XAllocColorCells
      (colormap_display,! Pointer to Display structure
       colormap,         ! Colormap id
       0,                ! T/F if planes are/are not contiguous
       &dummy,           ! Returns an array of plane masks; unused
       0,                ! Number of planes in the plan mask array
       colors,           ! Returns an array of color values
       colorcount))      ! Returns # of pixels in the pixel array
  {
    /* Ascertain that all colors were indeed allocated. */
    printf ("Could not allocate enough colors \n");
    exit (0);
  }

 /* Fill the color definition structure with CAC colors. */
 /* See figure 5.                                         */
 for (i=0; i<colorcount ; i++)
    {
     colorcell_defs[i].pixel = colors[i];    ! load a hardware
                                             ! colormap offset
     colorcell_defs[i].flags = DoRed | DoGreen | DoBlue;
                                         ! load CAC RGB components
     colorcell_defs[i].red   = rbuf[i]<<8;   ! shift red,
     colorcell_defs[i].green = bbuf[i]<<8;   ! green, and
     colorcell_defs[i].blue  = gbuf[i]<<8;   ! blue over 8 bits
    }
 /* Store the newly-filled colorcells into the     */
 /* hardware color map (Nye, 1988b). See figure 5  */
 XStoreColors
  (colormap_display, ! Pointer to Display structure
   colormap,         ! the color map to use
   colorcell_defs,   ! Color definition structures
   colorcount);      ! Number of XColor structures
```

Table 4. Statements To Modify The Hardware Colormap To Accommodate Cac Colors.

data.

ACKNOWLEDGMENTS

This work resulted from ongoing efforts within the NRL-SSC Map Data Formatting Facility (MDFF). The MDFF is in part funded by the Naval Air Systems Command (NAVAIR) F/A-18 (program element APN) and V-22 (program element 64262) programs. The authors thank the following program managers at NAVAIR for their support: CDR V.J. Chenevy, LCDR C. Cleaver, and Major J.P. Stevens (F/A-18); and CDR M. Redshaw (V-22).

REFERENCES

Defense Mapping Agency (1989). *Product Specification for ARC Digitized Raster Graphics (ADRG)*. Report PS/2JD/100. DMA Aerospace Center, St. Louis, MO. 1st Edition. April.

Lohrenz, M.C. (1991). *Military Specification. The Navy Standard Compressed Aeronautical Chart (CAC) Database.* NOARL SP 024:351:91. Naval Research Laboratory. Stennis Space Center. MS August

Lohrenz, M.C. and J.E. Ryan (1990). *The Navy Standard Compressed Aeronautical Chart Database*. NOARL Report 8. Naval Research Laboratory, Stennis Space Center, MS. July.

Nye, A. (1988a). Xlib Programming Manual for version 11. O'Reilly and Associates. Inc., Sebastopol, CA.

Nye, A. (1988b). Xlib Reference Manual for version 11. O'Reilly and Associates. Inc., Sebastopol, CA.

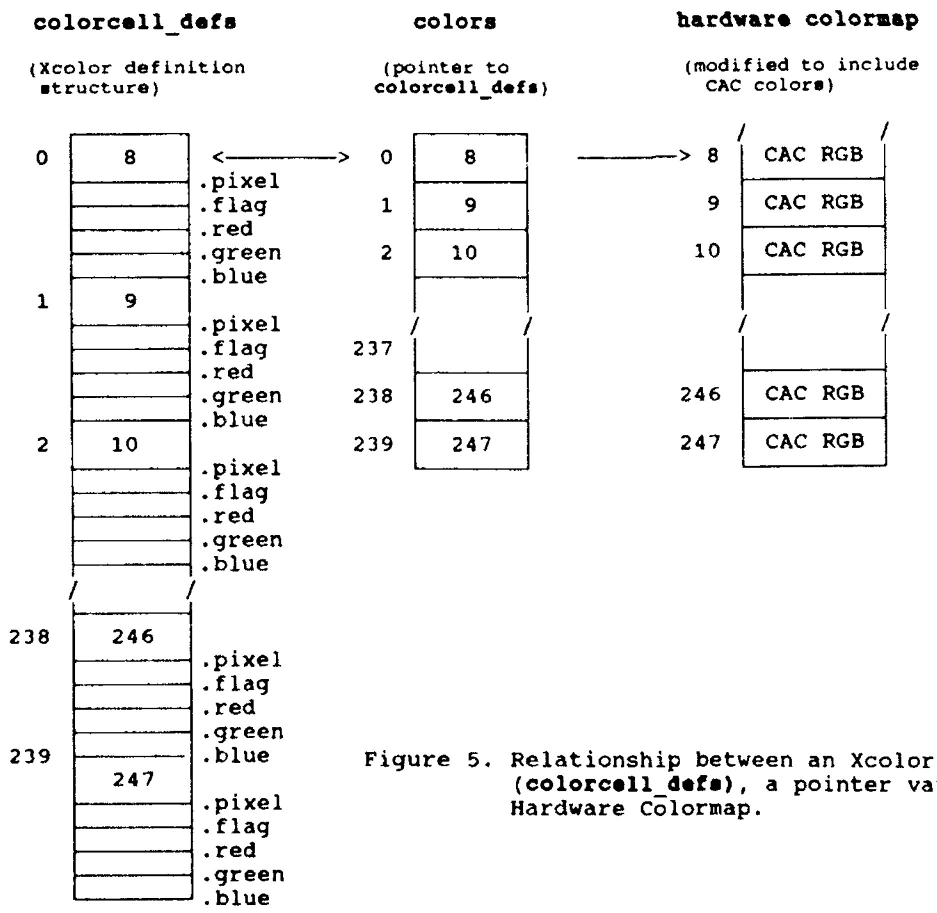

Figure 5. Relationship between an Xcolor structure (**colorcell_defs**), a pointer variable (**colors**) and the Hardware Colormap.